FPGA Implementation of Stereo Disparity with High Throughput for Mobility Applications

Carlos Y Villalpando, Arin Morfopolous, Larry Matthies
carlos@jpl.nasa.gov, arin@jpl.nasa.gov, lhm@jpl.nasa.gov
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

Steven Goldberg,
Indelible Systems, Inc, 8921 Quartz Ave, Northridge, CA 91311
indeliblesteve@gmail.com

Abstract—High speed stereo vision can allow unmanned robotic systems to navigate safely in unstructured terrain, but the computational cost can exceed the capacity of typical embedded CPUs. [1] [2] In this paper, we describe an end-do-end stereo computation co-processing system optimized for fast throughput that has been implemented on a single Virtex 4 LX160 FPGA. This system is capable of operating on images from a 1024x768 3CCD (true RGB) camera pair at 15Hz. Data enters the FPGA directly from the cameras via Camera Link and is rectified, pre-filtered and converted into a disparity image all within the FPGA, incurring no CPU load. Once complete, a rectified image and the final disparity image are read out over the PCI bus, for a bandwidth cost of 68MB/sec. Within the FPGA there are 4 distinct algorithms: Camera Link capture, Bilinear rectification, Bilateral subtraction pre-filtering and the Sum of Absolute Difference (SAD) disparity. Each module will be described in brief along with the data flow and control logic for the system. The system has been successfully fielded upon the Carnegie Mellon University's National Robotics Engineering Center (NREC) Crusher system during extensive field trials in 2007 and 2008 and is being implemented for other surface mobility systems at JPL.

Table of Contents

1. Introduction 1
2. The JPL Stereo Vision System 1
3. System Overview 2
4. Rectification Acceleration 3
5. Filtering Acceleration 4
6. Correlation Acceleration 5
7. Results 7
8. Summary and Discussion 9
Acknowledgements 9
References 10
Biography 10

1. Introduction

Image processing is becoming an important part of autonomous robotic behaviors for both terrestrial and space based rovers. A variety of sensors and applications are used in order for the robot to navigate and interact with its surroundings. One such example is a Stereo ranging camera system. In this system, a set of images are taken from a stereo pair of cameras and range information is computed from the captured images. Real time stereo ranging enables activities such as autonomous navigation and hazard avoidance. Currently, real time stereo ranging on large resolution images on the order of 1024 pixels square requires desktop class computers, but in many applications, power, size, and speed of available space qualified and/or embedded processors are a limiting factor. Flight qualifiable processor speeds in the gigahertz range are unavailable; instead, only processors in the low hundreds of megahertz are space qualified, making real-time stereo processing impossible for any reasonable image size. Given those restrictions, the use of Field Programmable Gate Arrays (FPGA) has become an increasingly attractive technique for embedded processing. Integrating a sequential processor to do sequential tasks, and FPGA fabric to do vector and/or parallel processing enables the low power and high computation ability required for robotic applications.

In this paper, we will discuss an implementation of a stereo ranging computation system. This system contains all the components needed for stereo computation. Most of those components, including image capture, are placed in the FPGA, while the final floating point range computation is placed in the host processor.

2. The JPL Stereo Vision System

Binocular stereo vision is a computationally expensive algorithm due to the large number of matrix operations. The current steps involved in the JPL stereo algorithm are: [1]

(1) Digitize the stereo image pair.

(2) Rectify the images

(3) Filter image using a bilateral subtraction filter to normalize the image pair and highlight features.

(4) Correlate, or measure image similarity by computing the Sum of Absolute Differences (SAD) for 7x7 windows over a fixed disparity search range.

(5) Estimate disparity by finding the SAD minimum independently for each pixel.

[1]978-1-4244-7351-9/11/$26.00 ©2011 IEEE.
[2] IEEEAC paper #1162, Version 2, Updated December 23, 2010

(6) Filter out bad matches by using the left-right-line-of-sight (LRLOS) consistency check.

(7) Estimate sub-pixel disparity by fitting parabolas to the three SAD values surrounding the SAD minimum and taking the disparity estimate to be the minimum of the parabola.

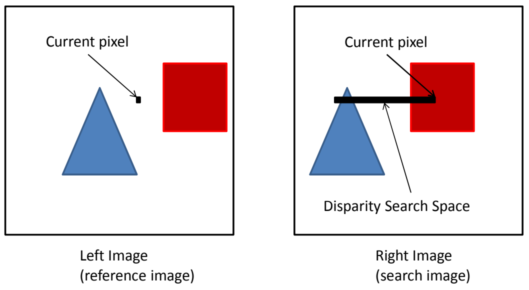
(8) Filter out small regions (likely bad matches) by applying a blob filter that uses a threshold on the disparity gradient as the connectivity criterion.

(9) Triangulate to produce the X-Y-Z coordinates at each pixel and transform to the vehicle co-ordinate frame.

Rectification determines a transformation of each image plane such that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes. Rectification reduces the correlation search space to a simple 1-D line based search. [1]

The image is filtered in order to remove the DC offset of each image. Filtering maximizes the correlation scores, producing better matches. If necessary, the image is then downsampled to the working size.

Correlation is the area based search for matching features between the two stereo pairs. One of the images in the stereo pair is defined as the reference image. For each pixel in the reference image, the other image is searched along the same scan line to find a match to the reference image. As a rule of thumb, the search space is chosen to be 10% of the image width and is called the disparity search space. The difference in pixel location of the best match between the stereo pairs is called the disparity for the reference image pixel. [1]



For a left-as-reference image, each pixel in the left image is compared against N disparity pixels in the right hand image. The search space starts at the same column as the left pixel column, and moves left. For a right-as-reference search, the search space in the left image also starts in the same column, but moves right.

Figure 1: Disparity search space illustrated

The LRLOS consistency check is a procedure that ensures that disparities obtained by choosing best matches along the left camera lines of sight agree with those obtained by choosing matches along the right camera lines of sight. If, for a given pixel, the left and right lines of sight do not give the same disparity, then the match is suspect, and is discarded. [1]

In this system, steps 1 through 7 are placed in the FPGA, while steps 8 and 9 are placed in the host processor. Steps 1 through 7 are very suited for FPGA implementation, as they are invariant and very data parallel. Step 8 currently yields little parallelization advantage, and step 9 uses floating point arithmetic, and may not be suitable to an FPGA.

3. System Overview

The JPL stereo vision system currently uses an Alpha Data ADM-XRC4 development board, which contains a Virtex 4 V4LX160 running at 66 MHz, 6 independent banks of 4MB Zero Bus Turnaround (ZBT) SSRAM, a PCI bus interface and a Camera Link interface. The key to the system layout is to maximize throughput by making each module fully pipelined, with each module feeding the next as much as possible. An overall dataflow and block diagram is illustrated in Figure 2 and is discussed below.

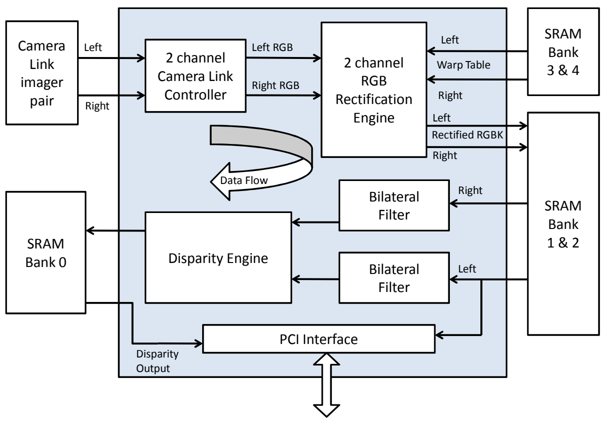


Figure 2: FPGA block diagram and data flow

There are two phases in the system, each 33ms long: The first is a pair of image capture modules which feed a pair of image rectification modules. The second is from rectified imagery through a pair of bilateral subtraction filters, which feeds a stereo correlator, which then produces disparity data.

The first phase is armed from a PCI bus register write, and triggers once the Camera Link module detects a new frame. The Camera Link cameras run at a 30Hz frame rate, and take just under 33ms (around 32 ms) to report the entire image.

Once triggered the raw imagery enters the system from the two Camera Link cameras and feeds one pixel at a time directly into each rectification module. The cameras feed 24 bit RGB data to the stereo FPGA system, while the FPGA modules generate a grayscale image to produce a 32 bit RGB plus grayscale pixel (RGBK).

The rectification module is a “linear in, random out” system, meaning that it accepts raw imagery in linearly from the top left of the image to the bottom right, but produces a rectified output pixel of random location depending on the rectification lookup table entry for that input pixel. This feature means that imagery can be acquired directly from Camera Link into the rectification module; however, the rectification module cannot feed the Bilateral Filter directly without buffering as the pixels would be unpredictably out of sequence. Therefore, the rectified image is written to an intermediary SRAM bank to save the warped imagery.

Once rectification is complete and the entire warped image is available in SRAM, an interrupt is issued over the PCI bus to signal the start of the next phase.

The host processor will require the use of the left rectified image color image for other machine vision tasks. In this system, the right rectified image is not used outside of Stereo. For efficiency, the left rectified image is read from SRAM bank 1 and simultaneously fed across the PCI bus to the host processor while feeding the first bilateral filter. The right rectified image is fed from Bank 2 in sync with Bank 1 into the second dedicated bilateral filter, and the combined left/right filtered imagery arrive together downstream at the stereo disparity module. This sequence is fully pipelined so that the first disparity pixels will arrive when about 17 rectified lines have been read into the filter. This phase ends when the disparity module outputs the final disparity pixel, and then the Camera Link module is re-armed for the next frame. This phase takes 28 ms, with 5 ms of dead time until the Camera Link module re-triggers. While the full 32 bit pixel from the rectified image is read by the host processor, only the grayscale component is fed to the second phase of the stereo system.

The PCI bus reads the disparity image for the host processor while the new Capture to warped imagery phase occurs. The ZBT banks are independent, with the disparity imagery in Bank 0 and the Rectification using Banks 1/2/3/4. This improves throughput. The entire disparity image will be read out in 11ms, 21ms before the last of the warped imagery is written to SRAM.

4. Rectification Acceleration

Rectification for stereo vision is the process of removing the distortion that a camera lens and imager system can apply to the world, and to correct any camera aiming errors so that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes. [1] Regardless of the quality of the lens and imager, there will be some distortion in the lens that can be eliminated by resampling the original image to one that would be generated by an ideal pinhole camera. Additionally, before performing stereo, the images need to be corrected so that the two images are aligned. No matter how well mounted the two cameras are they will be pointed ever so slightly in different directions. This can be corrected so that a feature in the left image is in exactly the same row as that same feature in the right image. This reduces the correlation search to a simple 1-D line based search.

Calibration images of targets with known geometries are used to generate a model of the camera’s intrinsic and extrinsic properties. That model is then used to create a large lookup table called a rectification table that describes the mapping from input image, to a final rectified image. The creation of this rectification table is only done once per camera set up.

The rectification table describes the projection of a pixel in the rectified image to a precise location in the raw image. The neighboring 2x2 pixels to the projected pixel are weighted according to the distance of the 2x2 pixels to the projected floating point coordinate, and the weighted pixels are averaged together to produce the rectified pixel output.

A 2x2 window in the raw image can influence zero, one or multiple output pixels. Each 2x2 input block of raw imagery has a rectification table entry which defines the location of its output pixels as well as the weights needed for each output pixel.

During run time this rectification table is used to quickly warp the incoming images to get rid of the lens effects and to get the two images pointed in the same direction, as described above.

As illustrated in Figure 2, the rectification module is broken into 2 independent channels for the left and right images and operates on true RGB 24-bit pixels.

A single channel consists of a FIFO link to the Camera Link Capture module, an SRAM port to read rectification table entries for the incoming pixels, a Control State module, a multiply-accumulate block to perform the averaging and weighting, and an independent SRAM port to write warped pixels out.

The rectified output pixel P_r is computed by from a bilinear interpolation of the 2x2 source window. The equation used for the interpolation is defined in Equation 1.[5]

$$P_r = \frac{(P_a(1-W_x)+P_bW_x)(1-W_y)+(P_c(1-W_x)+P_dW_x)W_y)}{256} \qquad (1)$$

Where the 2x2 raw pixels are P_a (top left), P_b (top right), P_c (bottom left) and P_d (bottom right), W_x is the weight along the x-axis and W_y is the weight along the y axis. Figure 3 graphically illustrates Equation 1. The dark dot represents the projected coordinate of the rectified pixel into the raw image frame.

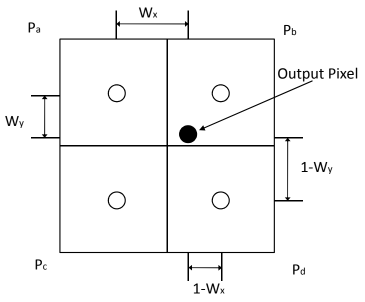


Figure 3: Rectification example

Each color channel is independently rectified by its own rectification module. A grayscale pixel is derived from the rectified RGB values and is appended to the RGB pixel to expand the 24 bit input into 32 bit output pixels.

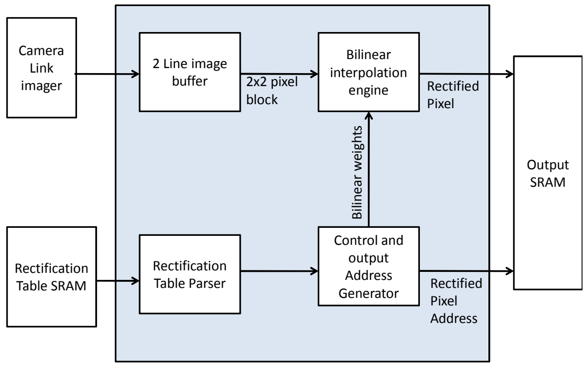


Figure 4: Single Channel Rectification Engine block diagram

5. Filtering Acceleration

The image filtering used in this system is the Bilateral Subtraction filter. [2][3] The amount of processing required is prohibitive for real-time applications on general purpose single CPU computers, and is extremely prohibitive in low power, embeddable CPUs for mobility applications. We have implemented this filter as a hardware module in order to accelerate processing to real-time speeds for large images and to complete another link in the chain for end-to-end stereo processing in an FPGA.

We have implemented a bilateral subtraction filter using a 9x9 pixel kernel in Verilog for implementation on a Xilinx Virtex4 family FPGA. The design has taken advantage of the ability to do many of the component computations in parallel with wide datapaths and limited off-chip communication, and the process is deeply pipelined so that the images are processed at the same rate the imager outputs image data. There are three parts to the filter, an image pixel pipeline with a 9x9 pixel window generator, an array of processing elements with a divider, and an image smoother and delay. A general overview is illustrated in Figure 5.

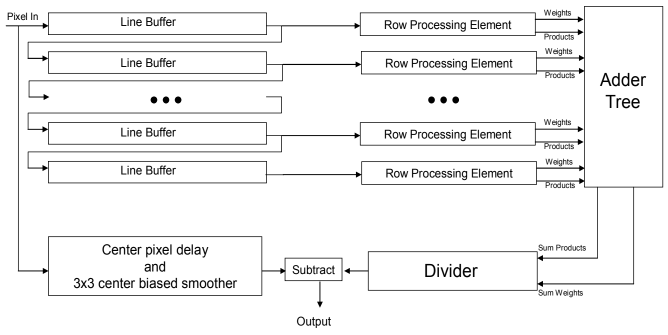


Figure 5: Filter Overview

For the second part of the filter, after the 9x9 window is extracted, the pixel data is fed to the processing elements. Each processing element is fed the pixel value for its position as well as the pixel value for the center pixel. An absolute difference is taken between those two values and the result is used as an address into a lookup table. Each processing element has a lookup table unique for its position in the window whose contents are the weight coefficients for the chosen Gaussian function for that position. The pixel value is then multiplied by the weight and the output of the processing element is the pair of the product and the weight selected. The products are fed into an adder tree to be added as well as the weights and those two sums are fed into a divider and used generate the bilateral smoothed image. The basic processing element is illustrated in Figure 6.

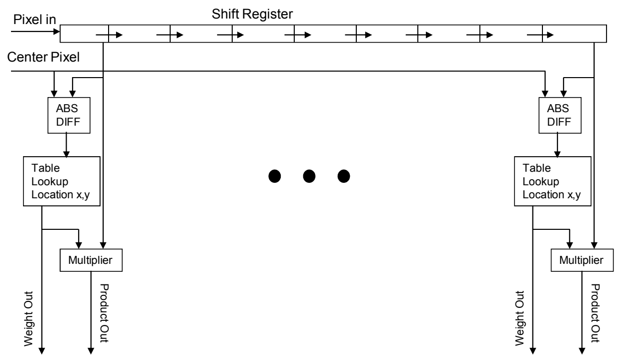


Figure 6: Filter Processing element block diagram

The third part of the filter is the image smoother and delay. The smoothing function is a simple 3x3 center weighted average. It is an average of the center pixel and the average of the surrounding 8 pixels. After the smoothing, the image is delayed by the processing time required for the bilateral smoothing. The bilateral filtered image is then subtracted from the 3x3 smoothed image to produce the output of the filter as illustrated in Figure 5.

6. CORRELATION ACCELERATION

SAD stereo correlation

The most demanding computations in Stereo come in the correlation and disparity search functions. This step involves performing a Sum of Absolute Difference (SAD) of a 7x7 window in one image against multiple 7x7 windows in the other image for each pixel. For example, for a 1024x768 image, and searching 10% of the image width, disparity requires 102 7x7 SAD operations per image pixel. This presents two tall requirements for a general purpose CPU, the first is the large number of arithmetic operations required, and the second is the large amounts of data that need to be moved in and out of the CPU. An FPGA can do many of the operations in parallel, and the ability to have extremely wide data paths limits the number of times the FPGA needs to go to off-chip memory to read data or to write intermediate results.

The next step is to find the minimum value of the 102 scores in both the Left and Right eye line of sights (LRLOS). In a sequential, non-vector processor, representative of many embedded processors, all these steps take a large amount of time to complete.[6]

The hardware is broken up into 3 major parts as illustrated in Figure 7. They are the score generator, a score delaying function necessary for LRLOS, and a disparity computer and checker. The overall hardware architecture emphasizes parallelism and pipelining in order to complete computations quickly. Each of these modules is fully pipelined with a defined latency and an issue rate of one data value per clock. Thus, the FPGA can produce one disparity output pixel per clock.

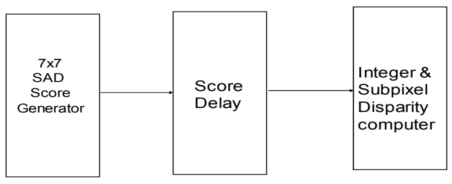


Figure 7: Disparity Engine Overview

The SAD Score generator consists of multiple absolute difference modules in running in parallel. For each stereo image pixel fed in, the left image pixel is applied against the 102 previous right image pixels. This result is fed to 102 parallel 7 pixel rolling sum calculators, (7RS) which results in 102 7 column by 1 row SAD scores representing the partial sum of the disparity search space for that pixel. A rolling sum of the previous seven values is used in order to cut down on the need to re-compute absolute differences for all 7 pixels in the window. This structure is illustrated by Figure 8.

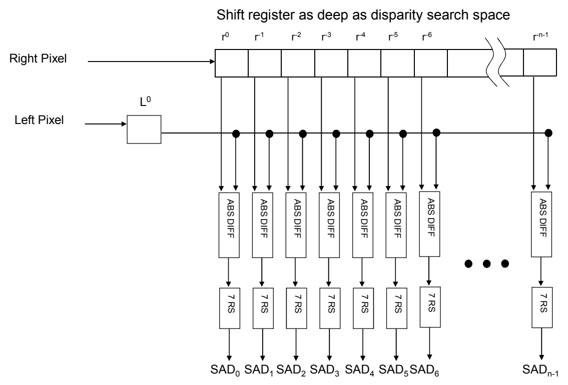

Shift register as deep as disparity search space
r0 r1 r2 r3 r4 r5 r6 rn-1
Right Pixel
L0
Left Pixel
ABS DIFF
7 RS
SAD0 SAD1 SAD2 SAD3 SAD4 SAD5 SAD6 SADn-1

Figure 8: 7 columns by 1 row SAD calculator

The output of the 7x1 SAD is fed into the 7x7 SAD score generator. The 7x7 SAD score generator operates on a similar principle as the 7x1 SAD calculator using a 7 value rolling sum. Instead of using shift registers, the 7x7 SAD computers use the Xilinx on chip memory primitives to store the previous seven lines worth of image data. That image data is fed into another 7x1 SAD calculator to re-compute the value that should be subtracted from the rolling sum. This method trades FPGA logic resources for memory, and reduces the amount of on-chip memory required over the method of storing the intermediate products of the 7x1 SAD score generator. 102 values are done in parallel as illustrated in Figure 9. The result of this module is a vector of 102 values representing the SAD scores for all 102 candidate disparities for one pixel.

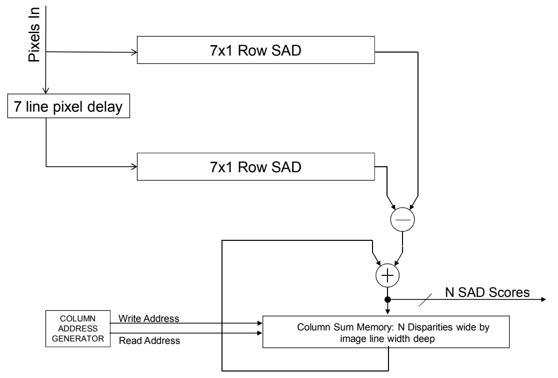


Figure 9: Seven by Seven SAD score generator

At this stage, we now have enough data to compute the disparity value for the current pixel; however, there are certain obstacle edge obscuration artifacts that give either degraded or false disparity data. We would also like to have a consistency check to determine if the computed disparity is valid. In order validate disparity, a check called the "left/right line of sight test" (LRLOS) test is applied. The primary disparity path is to check the left image pixel against 102 right image pixels. To perform the LRLOS check, the right image pixel is checked against 102 left image pixels, and the disparity computed from the right-image-as-reference is compared against the disparity computed from the left-image-as-reference. In order to prevent duplicating resources and computation, it is noted that if the results of the left image disparity computation are re-arranged, the values of the right image disparity computation can be obtained as illustrated in Figure 10. The implementation of the rearranging is a simple delay based on the corresponding disparity level. The left image score set is delayed by the latency of the re-arranger to keep left data synchronized with right data. The data is delayed as follows. If L(i) and R(i) are the left and right pixel respectively at position i, and D(x) is the SAD score at disparity x for pixel L(i), then the score at L(i)D(x) is equivalent to R(i-x)D(x).

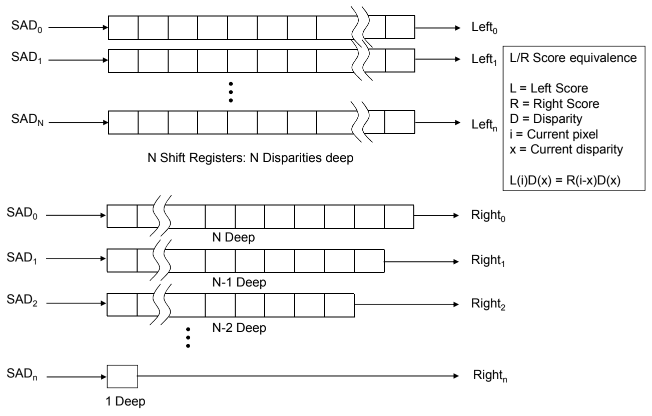


Figure 10: L/R Line of Sight Score Delay

We now have two vectors representing the left image SAD scores and the right image SAD scores. We can now compute integer and sub-pixel disparities. The left image is the primary image, and both integer and sub-pixel disparity will be computed from its data set. The right image is used only for the LRLOS check, and therefore only integer disparity is required. In addition to the LRLOS check, there are additional stereo consistency checks such as a min/max check and curve check. The overall architecture is illustrated in Figure 11.

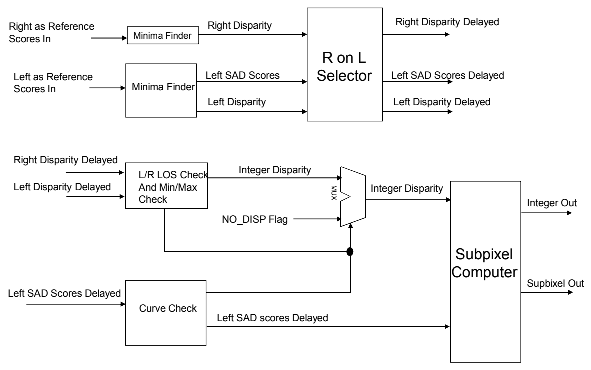


Figure 11: Integer/Subpixel Disparity Computer

Once we have the score vector for the current pixel, the first step in computing disparity is to find the minimum value in the SAD score array. The index of the minimum value will be the integer portion of the disparity for that pixel. The value of the minimum score of the left image vector, plus the score of the next and previous indices will be used to compute the fractional portion of disparity.

Figure 12 illustrates several stages of the pipelined minima finder. Each stage is repeated once per disparity searched. Each step checks the current disparity scores against the current minimum score. If the current value is smaller, the current disparity's index, score, and neighbor's values are saved. The same is done for the right image, however only the minimum value and index are saved, and only the minimum index value is output.

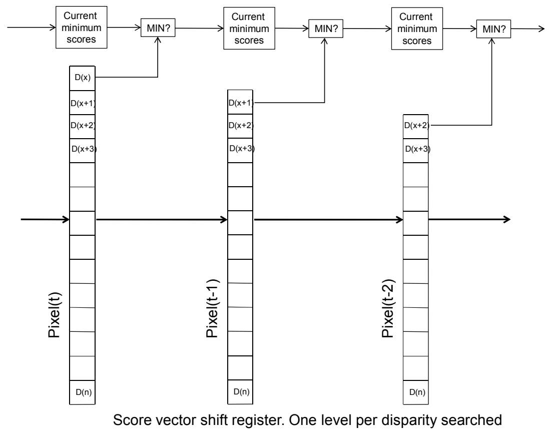


Figure 12: Minima Finder, repeated once per disparity

After the minimum is found, it is run through the stereo consistency checks such as minimum/maximum disparity check, minimum curve check, and LRLOS check. A maximal or minimal value for disparity is not a valid disparity, and the difference between the minimum score and its neighbors must be greater than a certain threshold for a suitable "curve" to the score vector.

The LRLOS check will compare the integer disparity of the left and right images, and if they differ by more than 1 index, it will fail the LRLOS check. It is an addressable Shift register. As disparities from the Right image are fed in, the address to read from the FIFO is derived from the current Left image disparity. For example, if the current left-as-reference disparity is 10, the right-as-reference pixel 10 pixels previous should also have a disparity of 10, plus or minus 1.

If any check fails, a no-disparity flag is inserted in place of the current pixel. After the checks have been performed, the data then goes to the sub-pixel computation element. The subpixel computation attempts to fit a quadratic curve to the minimum score, and its neighboring pixel's scores. Since the sub-pixel element requires a signed divide, a pipelined signed integer divider is used and the integer portion is added to the result after being delayed by the latency of the divider. The architecture is illustrated in Figure 13.

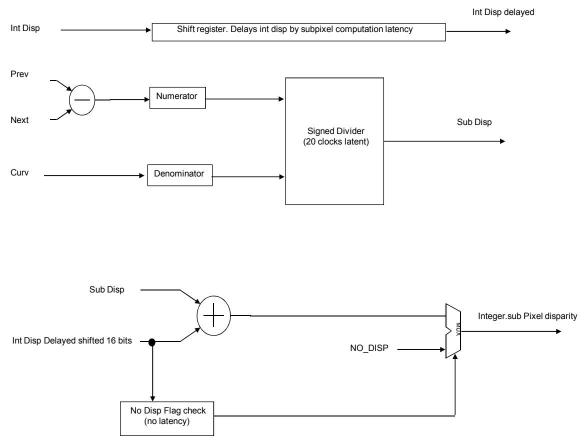


Figure 13: Subpixel Computer

The output of the subpixel computer is a 16 bit fixed-point word. There are 8 bits for the integer portion of disparity and 8 bits for the subpixel portion of disparity. At this point, disparity is complete, and the result is stored in memory.

SAD5 enhancement

There are certain artifacts in regular stereo correlation that blur or give false result at the edges of objects. To correct those effects, a method of combining SAD scores across multiple overlapping windows to produce the final SAD score for each pixel's candidate disparity called SAD5 was developed.[4] In SAD5, there are 5 scores generated for each candidate disparity. One score generated for the primary 7x7 window centered at the original pixel, and 4 more scores generated for the 7x7 windows for each pixel centered at each corner of the primary window. The lowest two scores of the 4 corner scores are added to the primary center score. The resulting score is then passed on to the disparity computation engine described in the previous section. To compute SAD5 in the FPGA, it is noted that all scores required to compute SAD5 are already computed by the 7x7 score generator described previously. A large shift register is placed in between the 7x7 SAD score generator and LRLOS score arranger. It is location in the flow is illustrated in Figure 14. This shift register is built from FPGA on-chip memory outputs the scores from the 5 overlapping SAD windows, and from those 5 outputs, a single output score per disparity is generated. Figure 15 illustrates the SAD5 score generator.

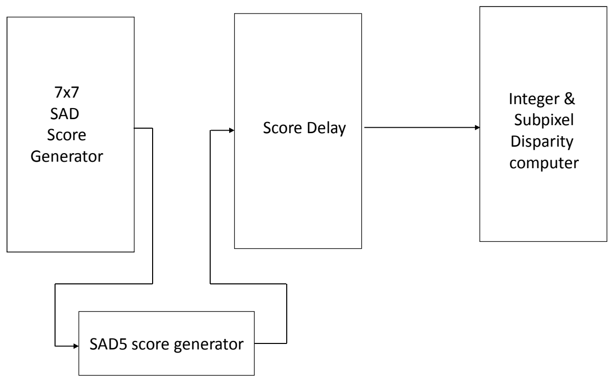


Figure 14: SAD5 score generator location

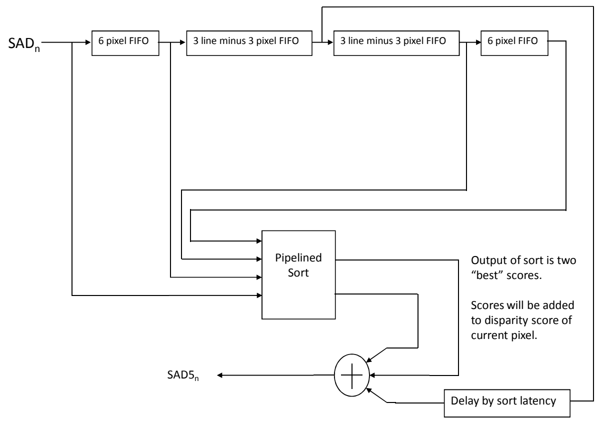

SAD$_n$
6 pixel FIFO
3 line minus 3 pixel FIFO
3 line minus 3 pixel FIFO
6 pixel FIFO
Pipelined
Sort
Output of sort is two "best" scores.
Scores will be added to disparity score of current pixel.
SAD5$_n$
Delay by sort latency

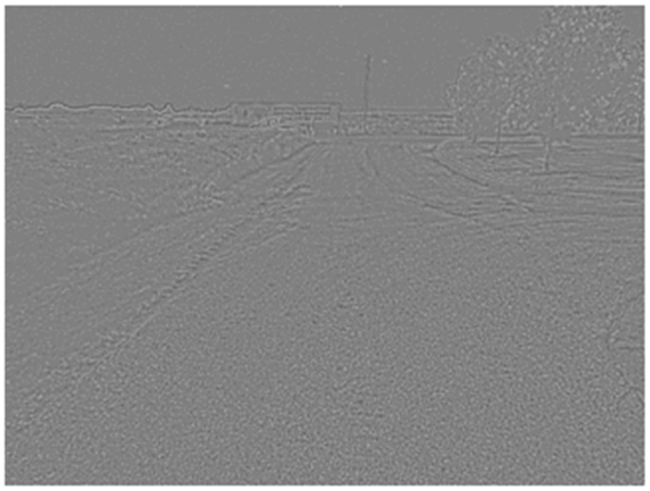
Figure 15: SAD5 Score generator (repeated once per disparity)

7. Results

The results of rectification, filtering and correlation have been independently compared against software equivalent modules and verified for performance and quality. An example data set in Figure 16 shows the input data and the outputs for each module.

(a) Raw Left Image

(b) Rectified Left Image


(c) Left Bilateral Subtraction Filter Image
(Contrast enhanced for print)

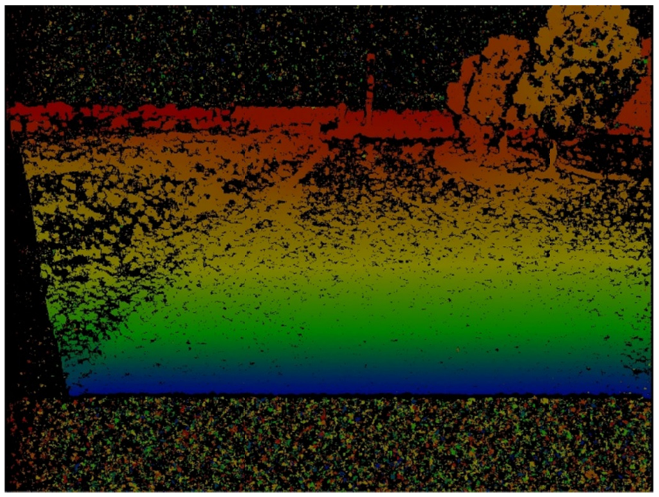

(d) Disparity Image

Figure 16. Note the slight curve in the raw input image (a), which is flattened in the rectified image (b). Also note that the Bilateral Subtraction Filter has small dynamic range (<30 intensity range per byte). The disparity image (d) has been colorized with black indicating no disparity, red meaning a small disparity value and blue indicated a large disparity value.

There is a large trade space between resource usage, performance and throughput.

The primary trade is on image size, FPGA resource use and throughput. Throughput scales linearly with image size. For the Rectification and Bilateral filter the resource usage is largely image size independent- they both are constrained by the kernel size of their computation, a 2x2 box and 9x9 box respectively.

The SAD correlation can be implemented as SAD1 or SAD5 with the change of a build time flag. Throughput is unchanged depending on the build type, but the SAD5 case uses roughly 4x BRAM that of SAD1 and uses an additional 4000 Slices (exact figures depend on image size).

Although not presented here, there is an optional method of reducing BRAM usage and another method for reducing Slices, both at the cost of reduced throughput for the SAD correlation stage. Using these methods BRAM can be reduced to 1/5 of previous use at a cost of 1/2 of previous throughput, and Slices can be reduced at most 1/10 of previous use at a cost of 1/10 of previous throughput.

Also not presented here, a Laplacian Box filter has been designed to substitute for the Bilateral Subtraction filter.

The Laplacian uses roughly 2x the BRAM and 1/5 the Slices of the Bilateral, at the cost of reduced quality of filtering results.

The standard build options are as follows:

FPGA Resource usage	BRAM[3]	Slices[4]
Virtex 4LX160	288	67,584
Virtex 5FX130t	576	40,960
Rectification	2	1,186
Bilateral Subtraction Filter	18	19,893
Correlator@1024 wide		
SAD1:	100	28,845
SAD5:	450	35,004
Correlator@512x384		
SAD1:	36	13,275
SAD5:	128	17,502
Example Complete Systems:		
Rect + Bil + SAD1 @ 1024x768	134	49,924
Rect + Bil + SAD5 @ 512x384	162	38,581

Throughput in this system is first limited by the Cameralink system, a fixed 30Hz, and then by the memory resources. With additional on-board memory the rectification and the filter steps could be interleaved so that the filter was running on the previous frame while rectification operates on the current frame. This would enable full 30Hz timing. The current system does not have enough room for such interleaving, and is thus limited to 15Hz at 1024x768.

If the Cameralink system was not used as the source of raw imagery into the FPGA board, but instead the PCI bus was used to load imagery in, the timing becomes more complex. In this case the throughput becomes dependent upon the image size, the PCI bus speed and the clock rate used on the FPGA. Our default design was 1024x768 imagery using a 32 bit, 66Mhz PCI bus and a 66 Mhz FPGA clock.

Using a PCI bus to load raw imagery results in an additional 43ms transfer time, resulting in a total frame period of: 43 ms (PCI transfer in) + 12ms (Rectification stand alone) + 28 ms (left rectified image read out over PCI bus and simultaneously through the filter) = 83 ms, or 12Hz. Disparity can be read out over the PCI bus while Rectification runs. Time can be reduced by: increasing the clock rate of the FPGA, improving the throughput of the PCI bus (such as by using PCI Express), reducing the image size, using grayscale instead of RGB imagery, or by using additional memory banks so the rectification and filter steps can be interleaved.

The maximum FPGA clock rate is 124MHz, which is constrained in this design by the internal performance of the SAD5 correlator.

As a comparison of the FPGA implementation to contemporary processors, the same algorithms for SAD1 implemented in the FPGA were run on a 1024x768 image pair on an Intel XEON 5160 at 3GHz, and a Core2 Quad Q6600 at 2.4GHz using the Intel's SSE2 instruction set for acceleration. The timing results for the two systems are as follows:

	XEON 5160 @3GHz	Core2 QUAD @ 2.4GHz
Rectification (2 images)	3 ms	6 ms
Bilateral Subtraction Filter (2 images)	6947 ms	8734 ms
Disparity	74 ms	87 ms
Total time	7024 ms	8827 ms

8. Summary and Discussion

This system is able to compute high resolution dense stereo at real-time rates with low latency in a small, low power package. A system capable of processing 1024x768 color images was fielded on an Alpha-Data development board placed in a small single board computer as a self-contained end-to-end stereo processor. The system can be tailored for various image sizes and disparity search ranges at synthesis.

The design of this system was from the beginning tailored to high speed ground mobility applications, and therefore the amount of on-chip resources used is commensurate with heavily parallel, deeply pipelined computation for maximum throughput. The image passes through each stage only once before being passed to the next stage. It is possible to reduce the throughput and save resources by making multiple processing passes on the image.

The ability to use the maximum amount of resources enables the computation of fast, high quality dense stereo disparity, freeing up any host processor for other tasks.

Acknowledgements

The work described in this publication was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration. This work was developed and matured under various NASA, DARPA, SPAWAR, and internal R&D programs.

[3] BRAM is 18Kbit

[4] Slice count is presented in the Virtex 4 Family's 4 input LUT reference frame, not the Virtex 5 Family's 6 input LUT reference frame for a 1-1 comparison

References

[1] L. H. Matthies, "Obstacle Detection for Unmanned Ground Vehicles: A Progress Report." Robotics Research, the 7th International Symposium. 475-486.

[2] A. Ansar, A. Castano and, L. Matthies "Enhanced Real-time Stereo Using Bilateral Filtering." In Proc. Intl. Symp. on 3D Data Processing, Visualization, and Transmission (2004)

[3] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in Proc. IEEE Intl. Conf. Computer Vision, pp. 836–846, 1998.

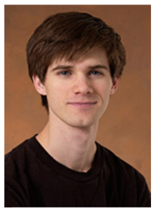
[4] H. Hirschmüller, P. Innocent, J. Garibaldi, "Real-Time Correlation-Based Stereo Vision with Reduced Border Errors," International Journal of Computer Vision, vol. 47, no. 1-3, pp. 229.246, April-June 2002.

[5] Donald B. Gennery. Calibration and Orientation of Cameras in Computer Vision, chapter "Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points", pages 123–136. Springer Verlag (A. Gruen and T. Huang, ed.), 2001.

[6] Goldberg, S.B. Maimone, M.W., Matthies, L, "Stereo Vision and Rover Navigation Software for Planetary Exploration", Proceedings, IEEE Aerospace Conference, 2002, vol. 5 pp. 5-2025 to 5-2036.

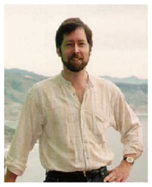
Biography

Carlos Y. Villalpando *is a Senior Member of Technical staff in the Advanced Computer Systems and Technologies group at the Jet Propulsion Laboratory. He is currently a digital designer for advanced computing techniques for machine vision applications in FPGAs as well as system designer and programmer for machine vision tasks on multicore processors. He earned his Bachelor of Science degree in Electrical Engineering, Computer Block at the University of Texas at Austin in 1996 and a Master of Science in Electrical Engineering-VLSI at the University of Southern California in 2003. He has been a member of the JPL community continuously since 1993 and has worked primarily on Technology development tasks.*



Arin Morfopoulos *is a member of the Robotic Actuation and Sensing Group at the Jet Propulsion Laboratory. He has been active in the FPGA design and implementation of vision algorithms on a half dozen DoD and NASA projects. He has been responsible for the system interfaces and integration of the FPGA vision algorithms on those tasks since 2007.*

Steve Goldberg *is a researcher at Indelible Systems in Los Angeles, California and Adigo AS in Norway. He earned a Bachelors of Science in Computer Engineering from the University of Southern California in 1999 where he was an undergraduate research assistant. Since he began contracting at JPL in 1997, he has developed real-time vision algorithms for autonomous navigation. He continues contracting for JPL and is currently working on agricultural robotics in Europe.*



Dr. Larry Matthies *obtained a Ph.D. degree in Computer Science from Carnegie Mellon University in 1989, then moved to the Jet Propulsion Laboratory, where he is currently a Principal Member of Technical Staff and supervisor of the Machine Vision Group. His research has focused on terrain sensing and obstacle avoidance algorithms for autonomous navigation of robotic vehicles. At JPL, he pioneered the development of real-time algorithms for stereo vision-based obstacle detection and he contributed to the development of the structured light sensor used by the Sojourner Mars rover. He has also developed algorithms for visual motion estimation from image sequences, 3-D scene reconstruction from image sequences, real-time terrain classification using multispectral imagers, and environmental mapping using sonar and stereo vision sensors. His group currently has research projects on computer vision for robotic vehicles sponsored by NASA, DARPA, and the U.S. Army; these projects include work on navigation of Mars rovers, asteroid and comet landers, and Earth-based robotic vehicles for urban and cross- country missions. He is a member of the editorial board of the Autonomous Robots journal and an adjunct member of the Computer Science Department at the University of Southern California. He has also been an invited speaker at the Frontiers of Engineering Symposium organized by the National Academy of Engineering*

.